On the Interplay Between Cyber and Physical Spaces for Adaptive Security

Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, *IEEE Fellow*, and Bashar Nuseibeh

Abstract—Ubiquitous computing is resulting in a proliferation of cyber-physical systems (CPS) that host or manage valuable physical and digital assets. These assets can be harmed by malicious agents through both cyber-enabled or physically-enabled attacks, particularly ones that exploit the often ignored interplay between the cyber and physical world. The explicit representation of spatial topology is key to supporting adaptive security policies. In this paper we explore the use of Bigraphical Reactive Systems to model the topology of cyber and physical spaces and their dynamics. We utilise such models to perform speculative threat analysis through model checking to reason about the consequences of the evolution of topological configurations on the satisfaction of security requirements. We further propose an automatic planning technique to identify an adaptation strategy enacting security policies at runtime to prevent, circumvent, or mitigate possible security requirements violations. We evaluate our approach using a case study concerned with countering insider threats in a building automation system.

1 Introduction

Computing and communication capabilities are increasingly being embedded into physical spaces, blurring the boundary between cyber and physical worlds [1], and offering novel attack opportunities to malicious agents [2]. For example, cyber-enabled physical attacks can occur when physical assets are cyber-controlled (e.g., digital access control to buildings). Similarly, physically-enabled cyber attacks can occur when physical access to assets enables cyber attacks (e.g., access to a particular computer may facilitate malicious access to digital information held on that computer). Understanding and managing the security threats that arise from the deployment of such cyber-physical systems (CPS) is a key challenge that is exacerbated by the interplay between the cyber and physical space (CPSp) that such systems inhabit. Although the literature is rich in accounts of security risk assessment methods (e.g., [3], [4]), these methods consider physical and cyber security separately [5] and are therefore unable to support the analysis of security threats arising from the interplay between the cyber and physical spaces that characterise a CPS operational environment.

In previous work [6] we advocated that the topology of cyber and physical spaces — their structure in terms of key elements and their relationships — can provide a system with both structural and semantic awareness of security relevant contextual characteristics. These include the location of assets being protected and the security controls that should be enacted in their proximity. Moreover, the location of potentially malicious agents can increase the threat of harm to assets in their vicinity. Discovering threats arising from the interplay between cyber and physical spaces suggests the need for an explicit representation of the topology of such spaces including their dependencies. In previous work [7] we investigated the use of the Ambient Calculus [8] to represent the topology of physical spaces to help identify and prevent potential future violations of security requirements. However, the Ambient Calculus encodes computation only as process-algebraic structural changes in a hierarchy and hinders reasoning about communication and links, key characteristics of cyber spaces. Furthermore, it does not allow reasoning about actions in the cyber space that are enabled by specific conditions in the physical space and vice-versa.

Changes in the topology, often triggered by movements of objects or agents in the physical or cyber space, can change a CPS attack surface dynamically. Although the primitives of change are well understood, their effect on the satisfaction of security requirements is not easy to predict as there can be complex sequences of actions leading to a violation. An adaptive security approach is needed to *discover* possible security threats determined by topological changes and then *counteract* those by enacting security policies to circumvent, prevent, or mitigate violations.

In this paper, we propose an approach to engineering adaptive security for CPS that exploits a formal model of the topology of cyber and physical spaces to discover and counteract at runtime threats that arise from topological changes. We do so by choosing to work with a form of *Bigraphical Reactive Systems* [9] as an underlying modelling formalism and providing novel *speculative threat analysis* and *adaptation planning techniques* to, respectively, discover and counteract security threats at runtime. Bigraphical Reactive Systems (BRS) can express the topology of cyber and physical spaces, their interplay, and the security requirements to be satisfied. We enable adaptation by maintaining a live model representation of the CPSp characterising a system's operational environment. We perform speculative threat analysis by utilising model checking for reasoning about the consequences that topological changes can have on the satisfaction of security requirements. We also propose a

- *C. Tsigkanos and C. Ghezzi are with Politecnico di Milano, Italy. E-mail: {christos.tsigkanos | carlo.ghezzi}@polimi.it*
- *L. Pasquale is with Lero, University of Limerick, Ireland. E-mail: liliana.pasquale@lero.ie*
- *B. Nuseibeh is with Lero, University of Limerick, Ireland, and with the Open University, Milton Keynes, UK. E-mail: b. nuseibeh@open.ac.uk*

planning technique to generate an adaptation strategy that prescribes what security policies to enact in order to prevent, circumvent, or mitigate possible violations of security requirements identified through analysis. To support the proposed analysis and planning techniques, we implemented a prototype verification tool. We motivate and illustrate our approach using a case study concerned with a modern bank branch that provides additional financial services through a local cloudlet [10] and monitored by CCTV and security guards. We evaluate our approach with respect to additional requirements formulated following discussions with an industrial partner concerned with countering insider threats. Our results demonstrate that our approach can identify and counteract emerging security threats arising from the interplay between cyber and physical spaces. The performance of our approach also confirms its applicability at runtime in adapting security policies of realistic systems.

The main contributions of this paper can be summarised as follows. We provide a novel approach to explicitly and formally represent the interplay of cyber and physical spaces in which systems operate. Based on this representation, we perform security analysis and planning to counteract discovered threats automatically and adaptively, by modifying security policies only when a security violation can occur. To the best of our knowledge, this represents an advancement with respect to traditional approaches, which are often developed based on fixed boundaries and assumptions. Our approach also has limitations. In particular, it does not account for inertia of the physical world in the selection and enactment of an adaptation strategy. Moreover, like other model-based security approaches in literature, we assume correctness of the model of the CPSp.

The rest of the paper is structured as follows. Section 2 describes our motivating example and Section 3 presents our approach for engineering topology aware adaptive security. Section 4 introduces BRS and how we adopt them to model the topology of cyber and physical spaces. Section 5 and 6 describe the speculative threat analysis and planning techniques, respectively. Section 7 discusses our evaluation results and Section 8 describes related work. Section 9 concludes the paper.

2 Motivating Example

To motivate our work on adaptive security for CPS we introduce an example used throughout the paper. The example illustrates how changes in the topology of the cyber and physical spaces that a CPS inhabits can lead to violations of security requirements.

A modern bank branch offers traditional counter services as well as advanced online services (e.g., investments, financial consulting) through an available cloudlet [10], a recent architectural innovation arising from the convergence of mobile and cloud computing. It provides fast remote cloud services to the mobile clients in its physical proximity, i.e. connected to the local wireless network. A cloudlet contains cached state from a remote cloud service and has high connectivity to it. Cloudlets are increasingly being used in diverse application domains, such as mobile commerce, healthcare and military defense. In our example, a connection from a customer's mobile device causes the provisioning of a new virtual machine (VM) on the cloudlet. The bank branch also instantiates a Building Automation System (BAS) comprising an infrastructure for controlling heating, ventilation, and air-conditioning (HVAC), as well as CCTV surveillance and lighting. The host manager located in the server room controls appliances (lights, HVAC, CCTVs), by sending commands to the network gateway that, in turn, forwards them to the target appliances. Additional devices include a server, desktop computers containing confidential documents, and a printer. As commonly found in a bank, safes containing valuable assets are placed in a safe room.

2.1 When Cyber Meets Physical

The operational environment of our example is shown in Figure 1. The physical space includes the structure of the building; the bank has a main area where counters are placed and from which it is possible to access the wifi area and a corridor. From the wifi area customers can connect to the local wireless network to use banking services. A customer connected to the network may request the provisioning of more than one VM. The cloudlet has a fixed capacity that for this example we assume to be four VMs. The wifi area delimits the physical area within which it is possible to connect[1]. From the corridor it is possible to access private areas of the bank, including a server room, offices, a safe room and a printer room. Figure 1 also shows the location of physical objects (e.g., cloudlet, safe) and agents (e.g., bank customers, security guards) in the space. For example, Eve and Alice who are bank customers are located in the wifi area, while the cloudlet, the host manager and the HVAC are placed in the server room. The topology of the physical space also describes proximity and reachability relationships. In this paper, proximity indicates whether entities are co-located; for example Mallory and Trudy, who are a visiting technician and a security guard respectively, are co-located in the same area. Reachability indicates whether an agent can access a physical entity; for example, Mallory can access the server room.

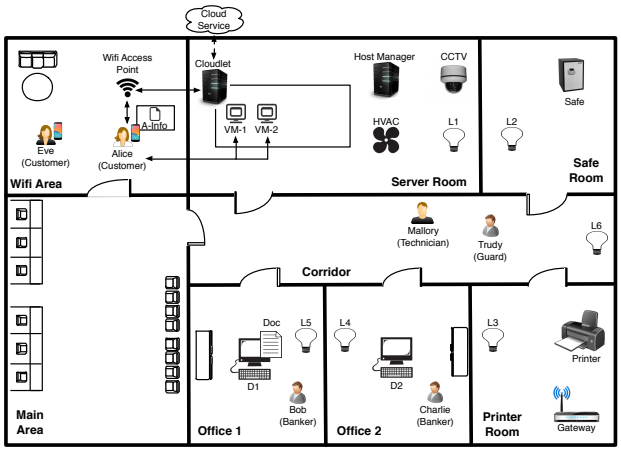


Fig. 1: Physical and cyber spaces of our example.

Like physical spaces, the topology of cyber spaces represents the structure of digital areas and the location of digital objects in such areas, such as bank account information or VMs. Some objects, such as mobile devices or the cloudlet,

1. Without loss of generality and for the sake of this running example, we assume wireless connections occur only from the wifi area.

can be conceived both as physical and digital entities. For example, Alice's mobile device delimits a digital area containing Alice's bank account information, while the cloudlet delimits a digital area containing a set of provisioned VMs. Proximity relationships indicate if two digital objects are placed in the same area (e.g., if two VMs are hosted on the same machine). Unlike in physical topologies, reachability relationships are also realised through communication links. For example, Alice's information can reach a cloudlet's VM associated with Alice (e.g., VM-1) because of the logical communication link between Alice's mobile device and VM-1. Although they are not shown in Figure 1 for reasons of clarity, network connections also exist between the gateway and the various digital devices, including ones belonging to the BAS (e.g., HVAC, CCTVs and lights).

Topological properties of a physical space can enable execution of actions in the cyber space. For example, the physical location of an agent carrying a mobile device in the wifi area allows her to connect to the wireless network and allocate a VM. Similarly, topological properties of the cyber space can enable execution of actions in the physical space. For example, connection of a digital device to the local network controlling BAS appliances allows sending commands to physical appliances (e.g., the HVAC) connected to the same network, possibly changing their behaviour.

The approach described in this paper aims at supporting the design of secure cyber-physical spaces. The designer can provide topological descriptions as well as specification of security requirements with respect to security goals (confidentiality, integrity, availability — CIA [11]). The following security requirements aim to protect digital (SR1 and SR2) and physical (SR3 and SR4) assets and operationalise such CIA security goals.

- *Availability of cloudlet services to connected customers*: if a customer is connected to the wireless network, then at least one the following statements must be valid: i) the customer has at least one VM allocated to her, ii) the cloudlet has not reached its maximum capacity (SR1).
- *Confidentiality of customers' information*: information transmitted by a bank customer to the cloudlet should never be received by other customers (SR2).
- *Integrity of the safe*: agents should always be accompanied by a security guard when they are in the safe room (SR3).
- *Integrity of the appliances*: appliances comprising the BAS should never execute commands originating from untrusted connected agents (SR4).

2.2 Adaptive Security in Cyber-Physical Systems

Different threats can arise dynamically when changes take place in the cyber or physical space. In this paper a security threat is conceived as a violation of a security goal [12].

Cyber threats can arise from changes in the cyber space and may cause harm to a digital asset. For example, if the cloudlet has reached its maximum capacity and Eve connects to the wireless network, the cloudlet service availability (SR1) can be violated since Eve's requests cannot be met. A security policy to counter this threat could enforce deallocation of a VM associated with another agent, or could oblige another agent or Eve to disconnect from the wireless network or leave the wifi area.

Physically-enabled cyber threats can arise from changes in the physical space and may cause harm to a digital asset. For example, assuming that Eve is in the main area, if she moves to the wifi area, she could connect to the wireless network and eavesdrop confidential information transmitted between Alice and the cloudlet, violating SR2. A security policy to counter this threat could prohibit unencrypted communications between customers and the cloudlet at the cost of increasing mobile device battery consumption, or could prohibit Eve from connecting to the wireless network.

Physical threats can arise from changes in the physical space and may harm a physical asset. For example, Mallory is an external visitor who is initially co-located with the security guard in the corridor. If she enters the safe room while the security guard remains in the corridor, the integrity of the safe could be violated. A security policy to counter this threat could oblige the security guard to move to the safe room to perform surveillance, or could prohibit Mallory from entering it.

Cyber-enabled physical threats can arise from changes in the digital space that can harm a physical asset, for example by subverting the functioning of appliances comprising the BAS which control heating, ventilation and lighting and perform surveillance. Note that protocols used in BAS (e.g., KNX [13]) do not include security features, because of limited resources or legacy deployments. Therefore, passwords employed to authenticate valid commands are sent in cleartext on the network, thus allowing key sniffing [14] and the possibility of actively subverting the correct functioning of appliances. In our example, to guarantee the integrity of the server, it is necessary to ensure the correct functioning of the HVAC located in the server room. To achieve this aim a security policy should prohibit execution by the HVAC of malicious commands originating from untrusted agents.

To support such adaptive security scenarios, the approach we advocate here consists of (1) monitoring topological changes in the operational environment, (2) identifying possible requirements violations that can arise from future topological configurations that may be entered by the CPSp, and (3) planning and enacting security policies that counter such possible violations.

3 Topology Aware Adaptive Security

Figure 2 provides an overview of our topology aware adaptive security approach. Adaptation builds on a live representation of the topology of the CPSp characterising a system operational environment modelled using BRS [9]. Adaptive security is achieved by implementing the activities of the MAPE (Monitoring, Analysis, Planning, Execution) loop [15]. Analysis and Planning are responsible for identifying possible security requirements violations and generating an adaptation strategy, respectively. Monitoring and Execution are responsible for enacting it at runtime.

During *monitoring*, events generated in the CPSp indicating the execution of agents' actions are received. For our example, such events can indicate access to a room by an agent, connection/disconnection of a mobile device to/from the wireless network, provisioning/deprovisioning of VMs on the cloudlet, and message exchanges between a mobile

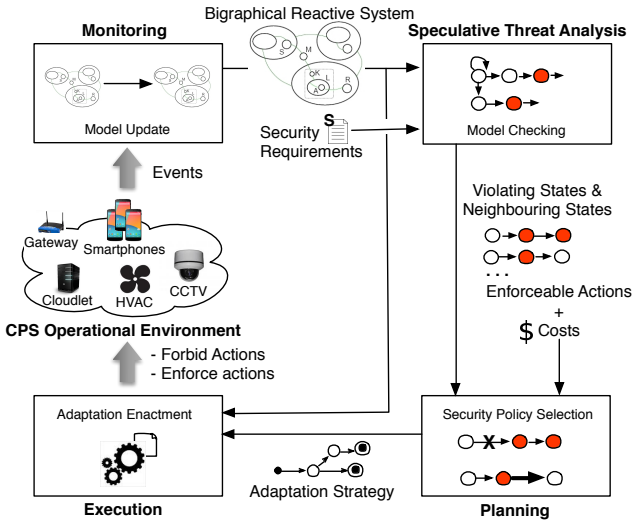


Fig. 2: Our topology aware adaptive security approach.

device and a VM. The model of the space maintained at runtime is also updated accordingly.

During *speculative threat analysis*, future topological configurations of the space representing violations of security requirements which can arise when agents perform actions are identified through model checking. The component performing analysis receives as input the model of the CPSp and the security requirements. Analysis identifies configurations where violations of security requirements take place, as well as additional neighbouring configurations of the CPSp before and after a violation.

During *planning*, an adaptation strategy that aims to counteract future violations of security requirements is generated. It consists of a set of security policies to be enacted at specific configurations of the CPSp. In this paper, a security policy can prohibit execution of actions leading to the violation of a security requirement, or could oblige (enforce) execution of actions to circumvent or mitigate a violation. This activity receives as input from the analysis configurations where security requirements are violated as well as their corresponding neighboring configurations. For each violation, it identifies a security policy if one exists, or prompts for a change in the design of the control system if the violation cannot be handled.

During *execution*, the adaptation strategy is enacted on the system. This activity receives as input the current configuration of the CPSp from the monitoring activity, and identifies if a specific state in the adaptation strategy is reached. If that is the case, it enacts specific security policies indicated in the adaptation strategy.

In this paper we focus on two critical activities of the MAPE loop: analysis and planning. We assume adequate monitoring is in place, as well as an execution environment through which selected adaptation strategies are enacted.

4 Modelling Cyber and Physical Spaces

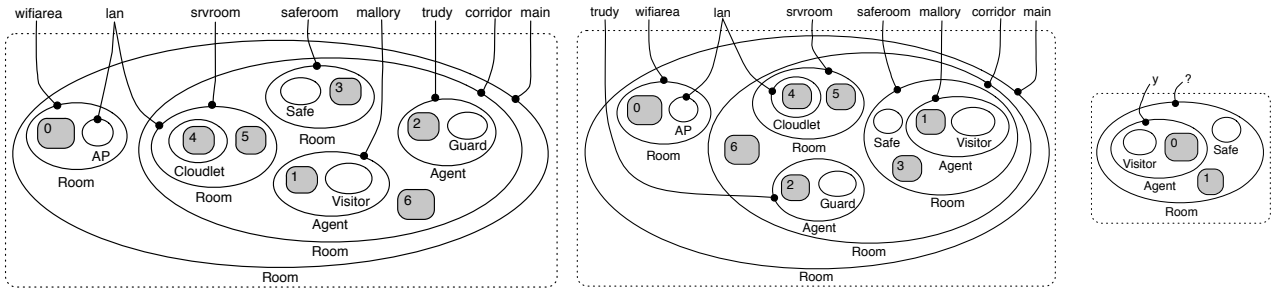
A modelling formalism for the topology of CPSp should allow the representation of the structure of those spaces and communication or linking among entities and agents. It should also enable reasoning about the effects of topological changes arising from (potentially concurrent) execution of actions by digital and physical agents. Process calculi are well established mathematical languages with well defined semantics that allow reasoning about properties of concurrent systems. They are essentially models of processes or agents interacting with each other and their environment. Several process calculi have been proposed, such as π-calculus [16], focusing on process migration, interaction and communication via dynamic channels, and assuming a flat process structure, or the Ambient Calculus [8], assuming a hierarchical process structure. However, such formalisms are not supportive of mechanisms to reason about topological characteristics of CPSp affected by both structure of the space as well as by links.

Bigraphs [9] are an emerging formalism for structures in ubiquitous computing. They consider both linking and structure. Bigraphical Reactive Systems (BRS) extend bigraphs with well defined semantics of dynamic behaviour expressed as a set of reaction rules. These rules allow reasoning about possible future topological configurations of the CPSp that are reachable from the current one, yielding a branching structure. In this section, we introduce bigraphs and BRS and explain how we have used them to model CPSp, their dynamics, and how requirements are specified.

4.1 Bigraphs and BRS

Bigraphs consist of two graphs. A *place graph* is a forest, a set of rooted trees defined over a set of nodes. A *link graph* is a hypergraph composed of the same set of nodes of the place graph and a set of edges each linking any number of nodes; this graph represents generic many-to-many relationships among nodes. Connections of an edge with its nodes are called ports. Place and link graphs are orthogonal, and edges between nodes can cross locality boundaries. Bigraphs allow us to express topological characteristics of CPSp; the place graph defines a hierarchical structure, which can model topological properties like containment, while the link graph can represent communication or some linking among nodes. What follows is a rather informal presentation of bigraphs as used in the scope of this paper; the interested reader can refer to the work by Milner [9] for complete definitions and proofs of the bigraphical theory.

Bigraphs can be described in algebraic terms (Formulae 1a-1e); in Section 4.2 we introduce an equivalent rigorous graphical representation. P, Q, and U are controls of bigraph nodes; controls are names that define a node's type. Nodes can be structured hierarchically through the containment relationship, expressed in Formula 1a. Two nodes may be placed at the same hierarchical structure level, as shown in Formula 1b. Additionally, bigraphs can contain sites, a special type of node (Formula 1c) that can be used to denote a placeholder; sites can be used to indicate presence of unspecified nodes. Each node control can be associated with a number of named ports. If a single instance node of a given type exists in the bigraph, the control uniquely identifies that node. Otherwise, we use a port name as a way to uniquely identify it. In Formula 1d the node identified by control K and port names in list w also contains U. Ports that appear in a formula with the same name are connected, forming a hyper-edge, called *link* in the sequel. In Formula 1e, W and R indicate different roots.



(a) Mallory is located with Trudy in the corridor. (b) Mallory is located in the safe room. (c) $\overline{SR3a}$.

Fig. 3: Bigraph configuration before (a) and after (b) a reaction is applied, leading to integrity violation SR3 (c).

$P.Q$	*Nesting (P contains Q)*	(1a)
$P \mid Q$	*Juxtaposition of nodes*	(1b)
$-_i$	*Site numbered i*	(1c)
$K_w.(U)$	*Node with control K having ports with names in w. K contains U*	(1d)
$W \parallel R$	*Juxtaposition of bigraphs*	(1e)

BRS support specification of *dynamic behaviour* by extending bigraphs with *reaction rules* defining possible reconfigurations. Reaction rules are parametric and specify how a bigraph can be modified by rewriting selectively some of its portions. Reaction rules have the general form of $R \rightarrow R'$, where R is a redex and R' is a reactum; both the redex and reactum are bigraphs. In particular, if a part of a bigraph that matches the redex is identified, it can be replaced with the reactum, in a fashion similar to graph rewriting. The use of reaction rules is illustrated informally in the next section through examples.

4.2 BRS to Model Cyber and Physical Spaces

The notions of node nesting (*containment*) and *linking* of bigraphs, can be used to model different aspects of a CPSp in a unifying way. The semantic association between these bigraph notions and the real-world phenomena they represent is not stated explicitly. We assume it is recorded separately as part of the model documentation.

Containment defines a hierarchical node structure, which can represent nesting of entities, like an agent residing in a room, or reachability relationships such as a room being physically reachable from another through a door. Additionally, it can be used to model a role held by some entity, by nesting a node representing a role in it. For example, the main room in Figure 1 can be modelled by a node that contains a node representing the wifi area; this indicates reachability of the wifi area from the main room. Likewise, the corridor node contains a node representing agent Mallory. Containment can also indicate the boundary within which an entity resides (e.g., a node representing Alice's digital information can be nested within a node representing her mobile device, to indicate the physical boundary within which the information resides).

Bigraph links are defined by associating nodes to specific port names; for CPSp modelling, this can be utilised in two ways: identifying instances of nodes, or representing some semantic relationship (e.g. connectivity) among nodes, by linking them to the same port name. We partition the set of port names by some criteria recorded as part of the domain model; e.g. set $\mathcal{A}$ for names of agents and set $\mathcal{R}$ for names of rooms.

Figure 3a shows a graphical representation of a bigraph configuration partially modelling the CPSp of our example. $Agent$ and $Room$ are examples of node controls signifying a node type. Containment is represented graphically by nesting a node inside another. $Room$ with name $corridor$, contains $Agent$s with names $mallory$ and $trudy$, as well as another $Room$ (the safe room). A token node $Visitor$ is contained in $Agent$ with name $mallory$, signifying that she is a visitor; similarly with the security guard. Sites (graphically represented by shaded boxes) denote the presence of other, unspecified nodes. For example, the fact that the $Room$ named $corridor$ also contains other entities besides the two agents and the safe room, is collectively represented by a site placeholder (6). By linking controls to names, ports are used to identify instances of controls (e.g., $mallory$ and $saferoom$); they are represented graphically as black bullets. Ports can also be linked together to form named edges; for example, edge lan between AP (access point) and $Cloudlet$ represents connectivity between them. Finally, the dotted outer box graphically represents the root. Using the algebraic notation, the same bigraph of Figure 3a can be represented as in Formula 2.

$$Room_{main}.(Room_{wifiarea}.(AP_{lan} \mid -_0) \mid Room_{corridor}.(\\ Agent_{mallory}.(Visitor \mid -_1) \mid Agent_{trudy}.(Guard \mid -_2) \mid -_6 \mid \\ Room_{saferoom}.(Safe|-_3) \mid Room_{srvroom}.(Cloudlet_{lan}.(-_4) \mid -_5))) \quad (2)$$

BRS and CPSp Dynamics

Reaction rules allow us to model the actions, whose occurrence in the real world cause the evolution of the CPSp over time. The following example models the configuration change due to an agent entering a room with which she is at the same hierarchical level. Sites in the redex can represent the matching of arbitrary nodes present in a specific part of a bigraph. In the reactum, nodes belonging to the matched subgraph will be placed in the positions of the corresponding sites, identified by the same number. Awareness of which entities are involved in actions executed in the CPSp must be reflected in a reaction specification. Moreover, the reaction rule must be expressed in a parametric way, to account for arbitrary agents and rooms. Thus, we specify along with the reaction, variables ranging over sets of port names that indicate the enactor entity (an agent in $\mathcal{A}$) along with other subject entities involved in the changes modelled

by the reaction (a room in $\mathcal{R}$). Use of variables is not strictly part of bigraphs, but was also proposed by [17].

$$[n \in \mathcal{A}, r \in \mathcal{R}]\, Agent_n.\, -_0 \mid Room_r.\, -_1 \mid -_2 \rightarrow Room_r.(Agent_n.\, -_0 \mid -_1) \mid -_2 \quad (3)$$

Formula 3 and Figure 4 show an $enter\_room$ reaction, where $Agent\ n$ moves into $Room\ r$, while all the other elements contained either in the $Agent\ n$ (signified by site $-_0$), around (site $-_2$) or in the adjacent $Room\ r$ (site $-_1$) are not modified. In the $enter\_room$ reaction, port n corresponds to the name of the agent, and port r corresponds to the room involved in the reaction. Note that the $enter\_room$ reaction can be applied to the configuration of Figure 3a (Formula 2), where agent named $mallory$ is at the same hierarchical level with the room named $saferoom$. After the application of the reaction, she is then contained in it, resulting in the configuration of Figure 3b.

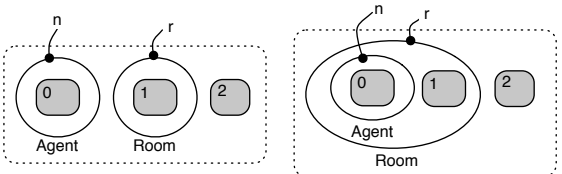


Fig. 4: (a) redex, and (b) reactum of the $enter\_room$ reaction.

Besides entering or exiting rooms, other reactions can be specified to model various actions that can be performed in the system. For example, bank customers can connect to the wireless network when they are located in the wifi area ($connect\_wifi$). They can also use the bank financial services through their mobile device, which must be connected to the wireless network; a new service request can cause the provisioning of a new VM on the cloudlet ($create\_vm$). Moreover, customers can transmit information tokens either in an unencrypted (tx) or encrypted (tx_enc) way. Given a bigraph that describes the initial configuration, the system evolves by applying reaction rules, which model the occurence of possible actions, generating new configurations. At each step, several applications of reaction rules may be possible, thus branching off possible new configurations.

4.3 Specifying Security Requirements for CPSp

A property for a given configuration can also be expressed by a bigraph. A configuration described by a bigraph C satisfies a property if the bigraph specifying the property can be matched against C, exactly in the same way a redex is matched against a bigraph to apply a reaction. Failure of matching the bigraph representing a property against C means, instead, that the property is not satisfied. The utilisation of sites in the bigraph specifying the property that is checked against C indicates that the portion of C that matches a site does not affect the property. To specify properties, we can quantify over the sets of port names which have been defined previously, e.g. for agents $\mathcal{A}$. We can also use the wildcard '?' is used to denote any port name. For example, an undesired configuration corresponding to the violation of the safe's integrity is represented in Figure 3c (and, equivalently, Formula 4a), indicating a visitor $Agent$ co-located with a $Safe$ in a $Room$:

$$\overline{SR3a}\text{:}\quad Room_?.(Safe \mid Agent_y.(Visitor \mid -_0) \mid -_1) : \forall y \in \mathcal{A} \quad (4a)$$

The configuration of Figure 3c (Formula 4a) matches the bigraph of Figure 3b. In the same fashion, we can specify an acceptable configuration where a $Safe$ is co-located with a $Visitor$ agent and a security $Guard$ agent (Formula 5a):

$$SR3b\text{:}\quad Safe \mid Agent_?.(Visitor \mid -_0) \mid Agent_?.(Guard \mid -_1) \quad (5a)$$

Having defined how to specify a property for a configuration, we can now show how system requirements can be specified. This is done by predicating on the branching structure induced by reaction rules utilising branching time temporal logic (CTL [18]). CTL includes two types of formulae: *state* and *path* formulae. State formulae are defined using the grammar specified in Formula 6, where φ is a path formula. Propositions a are expressed as bigraphs. Thus a state formula can be a bigraph configuration, the special proposition $true$, the composition ($\wedge$) of two subformulae, the negation ($\neg$) of a formula, and a path CTL formula prefixed by `E` (exists) or `A` (always) path quantifiers. `E` predicates that φ must hold on at least one path starting from the current state, while `A` asserts that φ must hold on all paths starting from the current state.

$$\Phi ::= true \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathtt{E}\varphi \mid \mathtt{A}\varphi \quad (6)$$

CTL *path formulae* are defined in Equation 7. A state CTL formula Φ prefixed by the *next* operator (`X`), and two CTL formulae Φ_1 and Φ_2 linked by the *until* operator (`U`) are valid CTL path formulae.

$$\varphi ::= \mathtt{X}\ \Phi \mid \Phi_1\, \mathtt{U}\, \Phi_2 \quad (7)$$

Path formulae are interpreted over paths of the branching structure. For example, given a path π, the property $\Phi_1\, \mathtt{U}\, \Phi_2$ is true if there exists a state s in the path that satisfies Φ_2 and each state that precedes s on the path satisfies Φ_2. Additional operators may be derived from the basic ones – $\mathtt{AG}\phi \equiv \neg\mathtt{E}\ true\ \mathtt{U}(\neg\phi)$ expresses global satisfaction of ϕ on all paths. Formulae 8a and 8b specify bigraphs $\overline{SR1}$ and $\overline{SR2}$, respectively, formalising topological configurations violating requirements SR1 and SR2.

$$\overline{SR1}\text{:}\quad Agent_z.(Phone_{wlan} \mid -_0) \wedge \neg VM_z \wedge Cloudlet_{lan}.(VM_? \mid VM_? \mid VM_? \mid VM_?) : \forall z \in \mathcal{A} \quad (8a)$$

$$\overline{SR2}\text{:}\quad Agent_z.(Phone_{wlan}.(Info_y) \mid -_0) : \forall z, y \in \mathcal{A}, z \neq y \quad (8b)$$

For example, $\overline{SR1}$ expresses the case in which an agent named z is connected to the network ($Agent_z.(Phone_{wlan} \mid -_0)$), no VM is allocated to her ($\neg VM_z$), and the cloudlet has reached its hardcoded maximum capacity ($Cloudlet_{lan}.(VM_? \mid VM_? \mid VM_? \mid VM_?)$). In Formula 9, a global system requirement is described; in every possible evolution of the initial configuration, requirements SR1-SR3 are never violated.

$$\mathtt{AG}(\neg\overline{SR1} \wedge \neg\overline{SR2} \wedge \neg(\overline{SR3a} \wedge \neg SR3b)) \quad (9)$$

5 Speculative Threat Analysis

Speculative threat analysis aims to identify potential violations of security requirements that can take place in future evolutions of the CPSp. To support it, we need to explore the state space of possible configurations that can be generated from an initial topological configuration. In this section we first illustrate how such a state space can be generated from

a BRS specification and formally represented as a state machine, where states represent configurations and transitions represent actions occurring in the environment that generate new configurations. We then discuss how the state space can be explored to check whether future configurations that violate the requirements can be reached.

Formally, the state machine we generate from the BRS specification to enable automated reasoning is a (Doubly) *Labelled Transition System* [18] (LTS) $\mathcal{L}$, defined as a tuple $\langle S, i, \Lambda, AP, \rightarrow, L\rangle$, where:

- S is a set of states describing configurations;
- $i \in S$ is the initial state;
- Λ is a set of transition labels;
- AP is a set of atomic propositions;
- $\rightarrow \subseteq S \times \Lambda \times S$ is a 3-adic relation of labelled transitions. If $p, q \in S$ and $\alpha \in \Lambda$, then $(p, \alpha, q) \in \rightarrow$ is written as $p \xrightarrow{\alpha} q$.
- $L : S \rightarrow 2^{AP}$ is a function that labels each state with the set of propositions that are true in that state.

A BRS specification can be interpreted as an equivalent LTS, which models the evolution of topological configurations caused by the occurrence of actions, modelled by reactions. Intuitively, given an initial configuration specified by a bigraph and a set of reaction rules, a target LTS can be generated by mapping bigraph configurations to states and the firing of reaction rules to transitions. The set of (AP) propositions that label a state (e.g., $p \in S$) can be systematically generated by declaratively encoding the corresponding bigraph configuration. LTS transition labels instead must indicate a specific instantiation of a reaction rule. To understand how new configurations are systematically generated and how the corresponding LTS transitions are labelled, consider an initial configuration specified by Formula 10, which represents two $Agents$ (a and b) outside $Room$s (p and k, respectively). The $enter\_room$ reaction (Formula 3) can be applied to this initial configuration leading to two possible ones, describing either $Agent_a$ or $Agent_b$ entering $Room_p$ and $Room_k$, respectively.

$$Agent_{\mathrm{a}}.-_0 \mid Room_p.(Agent_b.-_1 \mid Room_k.-_2) \tag{10}$$

When matching a reaction redex, the portion of a configuration that the redex matches indicates the specific bigraph nodes that instantiated the redex, including their named ports. Since ports are also used to indicate specific instances of node controls, each match can be distinguished by the ports of those nodes that matched the redex. Therefore, the transitions that lead to the states labelled with new configurations are labelled according to the reaction applied ($enter\_room$) as well as the named ports of controls of the initial configuration that instantiated the redex and appear in the involved entities specification of the reaction (in this case agents $\mathcal{A}$ and rooms $\mathcal{R}$). Thus, one transition will refer to $\{a, p\}$ and the other to $\{b, k\}$, indicating that $Agent_a$ entered $Room_p$ or $Agent_b$ entered $Room_k$, respectively. Formulae 11a-11b show the labels as well as the corresponding configurations generated. Matching can be automated by configuring existing approaches for BRS (i.e. [19], [20]).

$$\langle\{\mathrm{a}, p\}, enter\_room\rangle \quad Room_p.(Agent_{\mathrm{a}}.-_0 \mid Agent_b.-_1 \mid Room_k.-_2) \tag{11a}$$

$$\langle\{b, k\}, enter\_room\rangle \quad Agent_{\mathrm{a}}.-_0 \mid Room_p.(Room_k.(Agent_b.-_1 \mid -_2)) \tag{11b}$$

As the combination of a reaction name and associated ports identifying involved entities is unique since they correspond to actions in the CPSp, the derived LTS is deterministic, i.e. for every state p and transition label α, there is at most one state q such that $p \xrightarrow{\alpha} q$. Formulae 11a-11b show such generated labels along with new configurations that will be encoded in AP propositions in states.

The process of generating configurations and labelling states and transitions must be iterated by exploring all configurations. Whenever a reaction rule is applied to a configuration, a new LTS state is generated if no state already exists in the LTS that is labelled by the same configuration. This can continue until no new state can be generated, which corresponds to the case where the set of possible configurations is finite.

An exhaustive generation and analysis of all LTS states may not always be possible or may be inconvenient. The state space generated by the BRS can in fact be infinite, as for example in the case where allocation of new virtual machines in the cloudlet ($create\_vm$) is always possible. In other cases, although in theory the exhaustive exploration of the state space leads to a finite number of states, performance constraints may not allow generating or maintaining the full LTS, as it can grow to a size that causes memory or analysis performance issues. In addition, possible changes to the CPSp and hence to the BRS model would require performing again the entire (costly) LTS interpretation. Examples of such changes can be partitioning the main area to create two separate rooms or extending the maximum number of virtual machines that can be created beyond the limit initially specified in the BRS. In all above cases, state generation terminates when it reaches a predefined lookahead horizon, which corresponds to the exhaustive exploration of a certain number of subsequent actions. Overall, partial LTS generation aims to address scalability. Although a high lookahead horizon (even up to achievement of full LTS generation) allows in-depth exploration of the potential evolution of the CPSp configurations, it incurs increased space and time overhead. For this reason, it might be more suitable for design time analysis and for operational environments that rarely change.

Figure 5 shows an LTS fragment corresponding to the evolution of the example presented in Section 2. State a represents the initial configuration of Figure 1, where Alice is in the wifi area and her mobile device is connected to the wireless network. Eve is also located in the wifi area but her mobile device is not connected. From this state, Eve can connect to the network ($\langle\{eve\}, connect\_wifi\rangle$), or perhaps Alice can issue a new financial service request to the cloudlet causing the creation of a new VM ($\langle\{alice\}, create\_vm\rangle$). In state b Alice can transmit her personal information to one of her VMs hosted in the cloudlet ($\langle\{alice\}, tx\rangle$) or can dismiss one of the financial services requested causing the deletion of one of the VMs allocated on her behalf ($\langle\{alice, vm2\}, delete\_vm\rangle$). In state a Mallory and Trudy are in the corridor and, for example, Mallory can access the safe room ($\langle\{mallory, saferoom\}, enter\_room\rangle$).

To identify threats in future evolutions of the initial configuration (i.e. initial state of $\mathcal{L}$), we consider elementary bigraphical predicates as atomic propositions in a CTL logic,

which is interpreted over the states and branching paths of the transition system. In both modes - full or partial BRS interpretation - by ignoring transition labels and only considering state labels, the security requirements specification is verified against the LTS, discovering possible states that represent violations using standard CTL verification techniques, for which on the fly checking is also possible [21]. In Figure 5 for example, security requirement SR2 is violated in state f since information transmitted by Alice can reach Eve's mobile device. Similarly, if Mallory leaves the corridor and accesses the safe room, security requirement SR3 is violated (state c) since Mallory is co-located with the safe and without being accompanied by the security guard (Trudy).

6 PLANNING

The CPS can be viewed as a system composed of two entities: the environment, in which autonomous agents live and events occur, and the controller — the object of our design. This is a two-players game, where the environment is an opponent that may lead to unsafe states where requirements are violated, while the controller tries to counteract by devising a strategy able to prevent, circumvent, or mitigate such violations. The goal of planning is to compute an adaptation strategy to counteract potential threats, utilising violating states identified through analysis as well as information recorded on the transitions labels of the LTS. If the full LTS model is available, planning can be performed at design time and then enacted at runtime. However, if the LTS model is generated and analysed at runtime, as discussed earlier, planning also has to be performed at runtime, whenever possible threats are detected.

6.1 Adaptation Strategy Computation

To identify an adaptation strategy it is necessary to distinguish actions originating in the operational environment from those originating in the controller. Environment actions can be generally partitioned in two classes: *uncontrollable* (*U*) and *controllable*. Uncontrollable actions are those whose occurrence is out of the system's control. For example, the action of a person entering a room is uncontrollable if, for example, the door cannot be locked. Because such actions may lead to security violations, we can only attempt to mitigate their effect after they occur.

Controllable actions can be classified as *enforceable* (*E*), *preventable* (*P*), or both (*E*+*P*). We assume that all environment actions are specified by stating their class (*U*, *E*, *P*, or *E*+*P*). An action is *enforceable* by the controller if the controller can enforce its execution. For example, action $\langle\{alice, vm2\}, delete\_vm\rangle$ can be enforced by instructing the cloudlet to deallocate VM2. An action is *preventable* if its occurrence can be prohibited by the controller. For example, action $\langle\{alice\}, create\_vm\rangle$ can be prevented by the controller by prohibiting VM allocations by Alice. We assume that prohibiting or enforcing an action by the controller takes precedence over any other action performed by the environment and has an associated *cost*.

To express security policies, consider an LTS $\mathcal{L} = \langle S, i, \Lambda, AP, \rightarrow, L\rangle$ and V as the set of violating states found during threat analysis. An adaptation strategy identifies security policies to be enacted at specific LTS states representing configurations that lead to security requirements violations. We call these *alarm states*; a state is considered to be an alarm state if it is not a violating state and it leads to a violating state. Although our approach may consider alarm states at any distance from violating states, for simplicity herafter we assume a distance of one; that is, alarm states are connected by a transition to a violating state.

Whenever an alarm state is entered, the controller enacts a security policy by either (a) prohibiting actions that would lead to a violating state, or (b) pre-emptively enforcing an action that would correspond to exiting the alarm state and entering a safe (non-violating) state, thus circumventing the violation. If none of these security policies can be enacted, the environment may bring the system into a violating state. In such a case, the controller tries to enact a security policy that mitigates the violation, by enforcing actions that can bring the CPSp into a safe state. Because of the assumption on the priority of action enforcement over other environment actions, the sequence of actions that brings the CPSp to a safe state after entering a violating state can be considered as an atomic compound action.

Each security policy is specified as a set of security rules according to the OrBAC model [22]; $\texttt{is\_prohibited}(/\texttt{is\_obliged})\langle \texttt{s}, \alpha, \texttt{o}\rangle$ indicates that subject `s` is prohibited (resp. enforced) to perform action α on object `o`. Observe that we can deconstruct every transition label $\gamma \in \Lambda$ to its constituents $\langle \texttt{s}, \alpha, \texttt{o}\rangle$ where α is the name of the action associated with γ, `s` is the enacting entity, and `o` is the entity — if any — upon which the action is performed. For example, label $\langle\{trudy, saferoom\}, enter\_room, \rangle\}$ refers to subject $trudy$, action $enter\_room$ and object $saferoom$. Therefore, to simplify notation in the following we will refer to OrBAC terms `is_prohibited/ is_obliged` as operating on transition labels (or sequences of labels). We distinguish between three types of security policies; for each alarm state an optimal security policy is chosen, if any. In the remainder of this section we formalise and exemplify security policies.

A **prevent security policy** ($\mathfrak{P}$) aims at prohibiting the occurence of all actions that would lead to violating states from an alarm state (a). This policy can be selected if all transitions (γ) exiting a and entering a violating state are of class P (or $E + P$). Let $\mathcal{P}$ be the set of all transitions exiting from a and entering a violating state. Formally, the prevent security policy $\mathfrak{P}$ associated with the alarm state is defined as follows:

$$\mathfrak{P}(a) : \{\texttt{is\_prohibited}(\gamma) \mid \forall s' : a \xrightarrow{\gamma} s', \gamma \in P, s' \in V\}$$

The cost of the prevent policy is the cumulative cost of prohibiting the actions associated with the transitions in $\mathcal{P}$: $cost(\mathfrak{P}) = \sum(cost(\{\gamma \mid \gamma \in \mathcal{P}\})$. Conventionally, we assume $\mathfrak{P} = \emptyset$ and $cost(\mathfrak{P}) = \infty$ if the prevent security policy cannot be enacted in the alarm state.

A **circumvent security policy** ($\mathfrak{C}$) aims at circumventing violating states from an alarm state (a), by enforcing the execution of an action entering a safe state, if one exists. This policy can be devised by first computing the set of all transitions of class E (or $E + P$) exiting a and entering a safe state and then selecting among those the transition γ —if one exists— whose enforcement has minimum cost. The circumvent security policy $\mathfrak{C}$ associated with the alarm

state a is defined as the set that contains only one element $\texttt{is\_obliged}(\gamma)$; formally:

$$\mathfrak{C}(a) : \{\texttt{is\_obliged}(\gamma) \mid \exists s' : a \xrightarrow{\gamma} s', \gamma \in E, s' \notin V\}$$

We define $cost(\mathfrak{C})$ as the cost of enforcing the occurence of the action associated with γ. Conventionally, we assume $cost(\mathfrak{C}) = \infty$ if the circumvent security policy is not possible (i.e. no transition γ exiting an alarm state and entering a safe state can be found).

A **mitigation security policy** ($\mathfrak{M}$) aims at mitigating security requirement violations determined by the execution of an action that cannot be prevented by the controller. Intuitively, as an alarm state is entered a mitigation strategy first prohibits all actions corresponding to transitions that lead to violating states and are labelled as P (or $E + P$), if any. Then, if an uncontrollable action occurs that causes the system entering a violating state, the execution of a sequence of actions leading to a safe state is enforced. To formalise this security policy it is first necessary to introduce the following definitions. Given that $s_i \in S, 0 \leq i \leq n$ and $\alpha_i \in \Lambda$, a finite *computation* is defined as a finite composition of transitions:

$$s_0 \xrightarrow{\alpha_1 \cdot \ldots \alpha_n} s_n =_{def} s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots s_{n-1} \xrightarrow{\alpha_n} s_n$$

The concatenation $\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n$ of labels (representing actions) is called a *trace* originating from s_0. The sequence of states $s_1 \cdot \ldots \cdot s_{n-1}$ is called the *sequence of traversed states*. State s_0 is called the *originating state* of the sequence and state s_n is called the *end state*.

Let a be an alarm state and v a violating state reachable from a by a transition $\mathrm{a} \xrightarrow{\gamma} v$, where γ is a non preventable action. Let $\mathcal{EM}$ be the set of all traces such that (1) v is the originating state, (2) all transitions in the trace are labelled by actions of class E (or $E + P$), (3) all traversed states are violating states, and (4) the end state is safe (i.e., non-violating). This set is called the set of *enforceable mitigating* ($\mathcal{EM}$) recovery traces associated with γ:

$$\mathcal{EM}_\gamma(v) : \{\sigma \mid v \xrightarrow{\sigma} v_n, \sigma \in E^+, v \cdot \ldots \cdot v_{n-1} \in V, v_n \notin V\}$$

Among all $\mathcal{EM}_\gamma$ traces, we can choose the sequence of enforceable actions of minimum cost. This is performed for all non preventable transitions from the alarm state to violating states. In summary, a mitigating security policy $\mathfrak{M}$ is formally defined as a pair of sets $\langle \mathfrak{B}, \mathfrak{K} \rangle$. $\mathfrak{B}$ is a set of elements $\texttt{is\_prohibited}(\alpha)$, where α are all the transitions exiting the alarm state and entering a violating state that are preventable. $\mathfrak{K}$ is a set of pairs $\{x, \tau\}$, where τ is a sequence of $\texttt{is\_obliged}(\beta_1) \ldots \texttt{is\_obliged}(\beta_n)$ elements corresponding to the (minimum cost) $\mathcal{EM}$ trace $(\beta_1 \cdot \ldots \cdot \beta_n)$ preceded by transition γ whose corresponding action is x. Formally, the second constituent of $\mathfrak{M}$ associated with an alarm state a is defined as follows:

$$\mathfrak{K}(a) : \{\langle \gamma, \texttt{is\_obliged}(\mathcal{EM}_\gamma(v)) \rangle \mid a \xrightarrow{\gamma} v, v \in V\}$$

If no enforceable mitigating recovery traces can be found for each of the reachable violating states, no mitigating security policy exists for an alarm state. The expected maximum cost of $\mathfrak{M}$ is the cumulative cost of $\mathfrak{B}$ plus the maximum cost of the sequences of enforceable actions τ associated with the violating states that may be entered.

Consider the example previously presented where Mallory is located in the corridor; a fragment of the LTS generated is shown in Fig. 5. The system is in alarm state (a) and a requirement violation occurs if Mallory enters the safe room (state c). Suppose that the system is unable to control access to the safe room; it cannot forbid entering nor can it physically force Mallory to exit ($\mathcal{U}$ actions). However, it may be able to instruct the security guard to move inside the room to perform surveillance. In this case, a security policy will include rule $\texttt{is\_obliged}\langle \texttt{trudy}, \texttt{enter\_room}, \texttt{safe\_room} \rangle$. The adaptation strategy computed also includes prohibiting unencrypted transmissions when both Alice and Eve are connected to the wireless network (states e and g), as this is a preventable action and a circumvent or mitigation security policy is not applicable. The resulting strategy consists of the security policies to be enacted at alarm states e, g and a; they are shown as annotated dotted transitions in Fig. 5.

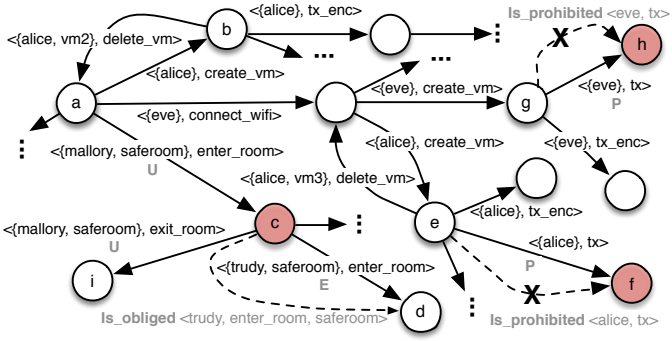


Fig. 5: LTS fragment (violating states are in dark).

In some cases planning can be unsuccessful, i.e. plans for violations in a given analysed alarm state do not exist and a model redesign is needed. For example, if the guard $trudy$ is absent (recall that $\langle \{mallory, saferoom\}, enter\_room \rangle$ is a $\mathcal{U}$ action), the CPSp will reach a violating state with no mitigating security policy to be enacted.

6.2 Adaptation Strategy Enactment

Monitoring and Execution are responsible for enacting an adaptation strategy at runtime through a controller mechanism [23]. During Monitoring, the events taking place in the operational environment are detected. Such events indicate the execution of actions represented as reaction rules in the BRS, and correspond to LTS transitions, which is maintained as a live model at runtime. Whenever an action is performed, the live model is updated reflecting the new state reached by the operational environment. If the new state entered is an *alarm* state, the corresponding security policy is enacted by Execution as a high-priority atomic activity, as stated in the adaptation strategy. We assume policies are enacted in a time interval that is less than the change rate of the environment, which is in turn related to its inertia; to satisfy this assumption we place the controller as an intermediary between the CPSp and the system.

Prevent or circumvent security policies will be prioritized; if these do not exist, mitigation security policies are enacted. If more that one option is viable, the one with minimum cost is chosen. If none is available, then the current design inevitably leads to a security requirement violation and must be revised. Called upon an *alarm* state a, Algorithm 1 checks if prevent or circumvent security policies have been identified. If such policies exist, the one with

minimum cost is enacted, prohibiting or enforcing actions as prescribed in the selected policy. Should none exist, the controller enacts a mitigating security policy by prohibiting actions prescribed in $\mathfrak{B}$, if any. Subsequently, it will block until a new event is received from the environment. If the received event represents the execution of an action x leading to a violating state, the controller will enact the rules enforcing the actions associated with sequence τ.

Algorithm 1 Monitoring & Execution

```
1: function EXEC(a, 𝔓_a, 𝔈_a, 𝔐_a)
2:     if 𝔓_a ≠ ∅ or 𝔈_a is defined then
3:         if cost(𝔓_a) ≤ cost(𝔈_a) then ENACT(𝔓_a)
4:         else ENACT(𝔈_a)
5:         end if ; return
6:     end if
7:     𝔐_a : ⟨𝔅, ℜ⟩ ; ENACT(𝔅)
8:     {x, τ} ∈ ℜ: ||block until x occurs in the environment||
9:     τ: ⟨enforcement plan for x ∈ ℜ⟩ ; ENACT(τ)
10: end function
```

As we observed, in certain cases due to performance constraints, analysis is unable to perform an exhaustive exploration of all possible configurations of the CPSp, and only covers a maximum number of subsequent system states up to a predefined lookahead horizon. In this case, as execution progresses, it may be necessary to extend the explored portion of the state space by updating the LTS model at runtime. Subsequently, analysis for possible violations is performed again, along with the security policy computation.

7 Evaluation

To support our approach and assess its effectiveness, we realised a prototype tool, implemented as a Python application[2]. Its functionalities were showcased [24] by implementing a test-bed instantiating a cyber-physical system similar to the example presented in Section 2.

To use our approach and its corresponding prototype tool, a security engineer has to specify the initial model of the CPSp, its dynamics in the form of reaction rules and security requirements using the bigraphical notation previously presented. Reaction rules corresponding to actions that can be performed by agents in the CPS are classified as P, E, $E+P$, or U. In the first three cases, the security engineer also estimates the cost of prohibiting or enforcing the action. In the example, actions include entering/exiting rooms, connecting and disconnecting digital devices, transmitting information, allocating VMs, as well as printing, opening and closing documents. Additionally, switching on and off appliances, such as lights and HVACs. The tool verifies satisfaction of security requirements. If violating states are found, it computes adaptation strategies for the corresponding alarm states, and prompts for a redesign if no security policies can be found. At runtime, adaptation strategies are enacted when alarm states are entered. In the rest of the section we assess the applicability of our approach by extending the example of Section 2 to formalise SR4 and consider three additional requirements (SR5-SR7), that we have formulated following discussions with an industrial partner working on access control. Finally, we assess space and time overhead necessary to generate an adaptation strategy and discuss results obtained.

2. The experimental prototype, specification and models of the example used can be found at http://178.62.206.217.

7.1 Applicability and Overhead

Our evaluation scenarios, formulated after discussions with an industrial partner, concern countering insider threats in a CPSp. In the configuration presented in Section 2, insider threats can be determined by agents exploiting their access levels to cyber and physical assets. For instance, a reasonable assumption is that connections to the network can be established at will, or that agents can physically move inside rooms in a trusted environment. Security controls deployed to counter such threats can be very complex as they must span both spaces, while not hindering overall system functionality.

Recall that the BAS should never execute commands originating from untrusted agents (SR4). In particular, SR4 states that the HVAC should be prevented from executing new commands until untrusted devices (e.g., $Phone$) carried by visitors are disconnected from the network gateway (Formula 12a). In a first scenario, we assume that the transmission of a potentially malicious packet ($PktHvac$) from a connected device cannot be prohibited; indeed checking the packet's source might not be satisfactory as this can be spoofed by an attacker. Therefore, the gateway may be enforced to conditionally discard packets received, until no untrusted device is connected to the network.

$$\textbf{SR4: } \mathtt{AG}(Agent_x.(Visitor \mid Phone_{wlan}) \Rightarrow \mathtt{A}\,(\neg Gateway.PktHvac\ \mathtt{U}\ Agent_x.(Visitor \mid Phone))) : \forall x \in \mathcal{A} \tag{12a}$$

In the second scenario, we assume that an employee is working on a confidential document (Doc) on desktop D1 (Figure 1). To guarantee the confidentiality of this document two security requirements must be satisfied (SR5 and SR6).

$$\textbf{SR5: } \mathtt{AG}(\neg((Agent_z.(Visitor \mid -_0) \mid Doc \mid -_1) \wedge \neg(Agent_y.(Employee \mid -_3) \mid Doc \mid -_4))) : \forall z, y \in \mathcal{A}, z \neq y \tag{13a}$$

$$\textbf{SR6: } \mathtt{AG}(Prnt.Doc \Rightarrow \mathtt{A}\neg(Agent_z.(Visitor \mid -_1) \mid Prnt.-_3)\ \mathtt{U}\ (Agent_y.(Employee \mid -_2) \mid Prnt.-_4 \mid Doc)) : \forall z, y \in \mathcal{A}, z \neq y \tag{13b}$$

To prevent a malicious individual from exploiting his permission to access Office 1 to see the document, SR5 states that Doc must not be opened on D1 while another $Agent$ with role $Visitor$ is in the office and an $Employee$ is not present. Note that when the document is opened, it is no longer contained in a computer but it is considered at the same hierarchical level of the agents and objects in a room. When the document is opened, two security policies may be enacted: i) prohibit access to the room, or when this is not possible, ii) enforce closing the document when a visitor enters the room. Similarly, SR6 states that the document cannot be printed until an $Employee$ is in the printer room. When the document is spooled to the printer ($Prnt$) through the network, the system chooses between two kinds of security policies: i) prohibiting document printing until an $Employee$ is in the printer room or, ii) prohibiting $Visitor$ agents from entering the printer room until an $Employee$ is inside to collect the printout.

Similar to requirement SR3, consider a requirement regarding the integrity of the assets placed in the server room, such as the cloudlet (SR7). To guarantee their integrity, it is necessary to ensure availability of the surveillance facilities in the server room; this prevents an agent from physically tampering with the equipment unnoticed. In particular,

while an agent is in the server room alone, the $CCTV$ should be functioning and the light $L1$ should be switched on, or alternatively a security guard should already be inside. To satisfy this requirement, access may be prohibited to the server room if the CCTV or the light are not functioning. If the CCTV or the light stop working while an agent is already inside the room, a security guard is obliged to enter.

$$\textbf{SR7:}\texttt{AG}(\neg(\neg(CCTV \wedge L1) \wedge (Room_{srvroom}.(Agent_{?}.(Visitor \mid -_0) \mid -_1) \wedge \neg Room_{srvroom}.(Agent_{?}.(Guard \mid -_2) \mid -_3))) \quad (14a)$$

We conducted our analysis on an Intel i7 (3.5GHz) processor; space and time overhead of the speculative analysis and planning of the configuration presented are shown in Table 1. We compare the overhead determined by analysing (identifying violations) and planning (computing adaptation strategies) for an LTS of increasing lookahead horizons (e.g., 2, 4 and 6) with the one determined by full interpretation. Results of Table 1 do not include LTS generation. The execution time of analysis and planning increase proportionally with the space overhead.

	# States	# Trans	Analysis Time	# VStates	Planning Time
L 2	93	171	3 sec	31	0.1 sec
L 4	1026	3703	22 sec	506	0.6 sec
L 6	4807	24002	198 sec	2816	25 sec
Full	24001	195613	~8 min	18008	~18 min

TABLE 1: Space and time overhead of Analysis & Planning.

7.2 Discussion

We have demonstrated that BRS are a suitable formalism to represent the topological properties of CPSp and reason about their interplay. Moreover, by utilising CTL properties over bigraph configurations as propositions, we were able to encode and verify even complex security requirements (e.g., SR4-SR7) predicating on execution paths. Guaranteeing the satisfaction of such requirements allows us to enforce fine grained behaviour, such as conditional access depending on past actions (e.g., for authentication purposes).

Our results highlight advantages and disadvantages of generating the full evolution of the BRS model compared to bounding analysis to a specific lookahead horizon. Although the time to perform analysis when the state space is not explored fully is lower (especially for a small lookahead), the adaptation strategy must be regenerated frequently at runtime. Instead, although the time to perform the analysis when the full model is adopted is higher, a new adaptation strategy will be regenerated sporadically, only when an exogenous change takes place, or at design time.

The reference implementation used is an experimental, unoptimized prototype; state of the art tooling would reduce overhead by several orders of magnitude. Moreover, strategies reducing the number of reaction rules considered during LTS generation could be investigated further. For instance, correlating undesired configurations with domain-specific information such as existing access control policies, would allow us to ignore states violating access control policies during analysis. Another possibility is ignoring reaction rules in the BRS model that do not have any impact on the satisfaction of the security requirements. However, our preliminary results do demonstrate the applicability of the approach in its context, and encourage further investigation.

As with any model-based approach, guarantees of completeness are in the scope of the model specified by the security engineer. Our approach does not take into account the inertia of the system, i.e. the time necessary to execute an action, which is especially relevant to physical systems. To address this limitation, in future work we will encode metric temporal characteristics into the underlying logical framework used. This will enable the selection of a security policy also depending on the time necessary to enact it. In this paper security policies are assumed to be enacted as an atomic action, occurring before the environment can change spontaneously. Another assumption made in our approach is that events observed correspond to the correct actions taking place in the CPSp; this is obviously a limitation.

8 Related Work

Our work touches a number of related areas and makes a novel contribution in software engineering of adaptive security for CPS. First we review existing work that adopts BRS to model and reason about CPS. Second, we discuss related approaches that use cyber-physical test-beds and formal methods to engineer secure CPS. Finally, we compare our work with existing adaptive security approaches.

8.1 BRS for Modelling Cyber-Physical Spaces

The adoption of BRS for modelling has been considered extensively in literature. Walton et al. [25] focused on on BRS as a formal model to represent indoor spaces and mobility of objects and agents in those spaces; this work aims at reasoning about path-based navigation tasks. Our group [26] has used BRS to provide formal semantics to Building Information Models, a standard for modelling physical spaces, in order to support verification of properties of smart buildings. In contrast, in this paper we use BRS to select security policies, where hierarchical structures are considered along with linking between CPSp entities.

BiAgents [27] are a formalism which encompasses bigraphs for modelling the physical space and abstract algebraic structures for the cyber space. Unlike our approach, which explicitly aims to model the interplay between cyber and physical spaces, a BiAgents model clearly separates them, and their interaction is fairly limited. BiAgents have been specifically used to identify strategies to prevent ill-defined concurrency behaviours that can emerge from cyber agents operating in shared physical structures. However, they are not suitable for identifying security breaches that can arise from the interplay between cyber and physical spaces. BRS with sharing [19] are used by Calder et al. [28] for modelling and reasoning in network topologies and management systems. Similarly to our approach, BRS are used to perform analysis at runtime in order to verify access control policies when network events occur. However, in this work BRS have not been used to model and reason on the interplay of physical and cyber spaces, and analysis results are not used to select security policies to counter identified violations. Benford et al. [17] use BRS to model socio-technical and human behaviour aspects alongside system behaviour and reason about their interplay. BRS are extended with probabilistic events, analysis of reaction rules (e.g., for verifying invariants) and a pattern system. Similarly to our work, bigraphs are used as propositions

in a logic and variables express reaction rules over multiple bigraph names. However, although we utilise similar modelling principles of the BRS formalism, we focus on the realisation of a complete methodology and framework that ranges from modelling, analysis and automated, adaptive generation of security policies.

The cornerstone underlying BRS (and thus also enabling reasoning on CPSp) is the matching problem [9], i.e. conditionally associating bigraphs and generating new instances. Apart from the closely related fields of graph transformation, term rewriting or term graph rewriting, we refer to existing work that has specifically approached bigraph matching. Several approaches for operating on various forms of bigraphs exist and can in principle be configured for the matching constituent of the approach outlined in Section 5. A SAT based algorithm for the matching problem [29], and bigraphs with sharing where the place graph can be a DAG are developed in [19]; a complete bigraphical framework is realized. BPLTool [30] enables reasoning on BRS systems with binding, and has an emphasis on implementation correctness with a matching system based on formally defined inference rules [31]. A similar approach has been followed in DBtk [32], a tool for reasoning on directed bigraphs, a form of bigraphs with directed edges. However, this work does not address checking properties of bigraphical models. SBAM [33] implements a variant of stochastic bigraphs, pure BRS equipped with stochastic semantics. The form of non-binding bigraphs that we use in this paper is similar, in that all controls are considered active, and there are no inner names. BigMC [20] is a model checker for bigraphs; from a description of a model, it finds all possible future configurations of this model while checking a specification expressed as a state matching property against them. The term language we use draws heavily from BigMC's. In our realization we perform checking of CTL properties whose propositions are bigraphical predicates, and support specifications with variables to describe reactions and properties.

8.2 Secure Cyber-Physical Systems

Some research in security of CPS has focused on assessing the effects of cyber threats against both the physical and the cyber dimension of networked critical infrastructures [34], [35] by using existing cyber-physical test-beds. These approaches focused on modelling dynamical systems such as sensors or actuators signals, for which timing concerns are fundamental. Our work instead aims to identify threats determined by changes of the cyber-physical space. and can be directly applied in practice in all cases where time needed to enact security policies is negligible with respect to timing of events occurring in the physical world. This is the case in access control in smart buildings, where actions such as locking doors, controlling network connections or access to services can be performed in a negligible time.

Similarly to our work, other approaches are grounded in formal methods to identify violations of security requirements in CPS. Akella et al. [36] propose an approach to detect confidentiality breaches determined by the composition of physical systems with cyber components. Sensitive information about a physical component can be inferred through behavior observation about related cyber components. A composite model of the CPS is formalised using the Security Process Algebra [37] and confidentiality properties are expressed as bisimulation-based non-deducibility – low level events observations from physical elements should not be affected when these are composed with cyber components. Automated detection of confidentiality breaches is performed by using the CoPS model checker [38]. A threat model for CPS [39] has been proposed to incorporate typical characteristics of physical systems, such as survivability to abnormal behavior and the possibility to recover after critically vulnerable states are reached. A CPS is modeled as a finite, hybrid timed automaton with faults. An adversary can exploit a certain set of transitions related to system vulnerable states. The framework is adopted to identify transitions whose execution must be forbidden and to provide formal proofs of protocols. Dimkov et al. [40] discuss insider threats that span physical, cyber and social domains. They build upon KLAIM [41] dialects for agent interaction and mobility to formalise insider threats and provide a framework to identify attack scenarios. However, none of the aforementioned approaches can automatically suggest security policies to counter security requirements violations. Furthermore, as far as we are aware, model checking techniques have not been used at runtime to discover security breaches determined by CPSp topological changes.

Extensions to the role-based access control (RBAC) model have been considered to include topological features into policy decisions. In particular, Prox-RBAC [42] proposes a policy language and an enforcement architecture to specify and enforce constraints according to agents physical proximity. This view of proximity originally restricted to spatial concerns was extended [43] to include characteristics belonging to the cyber and social spaces such as agents attributes (e.g., similar roles), social networks (e.g., short number of hops in a social network), communication channels, and time. Although a richer notion of proximity, including agents' attributes, social networks and time is relevant and can be incorporated in our approach, Prox-RBAC does not allow security analysts to define policies depending on additional topological relationships such as containment and reachability. Furthermore, it does not allow specifying complex temporal constraints describing potential evolutions. Finally, in [43] constraint satisfaction is checked when the permission to perform an action is requested and while an agent assumes an active role; this does not allow an optimal adaptation strategy to be selected in advance.

8.3 Adaptive Security and Enforcement

As far as we are aware, existing research on adaptive security [44] has not focused on CPS, which can be targeted by multi-vector attacks exploiting vulnerabilities of both cyber and physical components. Salehie et al. [45] propose a requirements-driven approach for dynamically re-estimating the risk of harm, depending on assets and context changes. Security threats, attacks and vulnerabilities are modeled in advance, and predetermined security controls are adjusted at runtime depending on the varying risk of harm. Architecture-based self-protection [46] aims to detect and mitigate security threats based on an architectural representation of the software that is kept in sync with the running system. The model provides information related to the impact of security breaches on the

system and allows engineering security controls by applying specific architectural design patterns. However, these approaches [45], [46] are based on the assumptions that security controls are predetermined and vulnerabilities are determined by individual system components. In previous preliminary work [7], we investigated the use of Ambient Calculus [8] to model the topology of the physical space and perform speculative threat analysis to reason about the impact that changes in the topology of the physical space can have on the satisfaction of security requirements. We decided, however, to abandon this formalism because it does not provide adequate support for modelling and reasoning about the interplay between cyber and physical aspects. The properties that correspond to the requirements in our approach are encoded as system safety properties, enjoying results on enforcement mechanisms pioneered by [23]. Notions of controllable and observable environment actions are formally treated in [47], where decidability of enforceability is studied. We further distinguish controllable environment actions in enforceable and preventable. Similar classifications of actions are common in other areas such as supervisory control theory (e.g. [48]). In practice, enforcement of the adaptive security policies generated in this paper can be achieved by configuring edit automata [49].

9 Conclusions and Future Work

In this paper we proposed an approach for engineering topology aware adaptive security for cyber-physical systems, focusing on the interplay between cyber and physical spaces characterising the operational environment. We used a form of BRS to represent cyber-physical spaces and their dynamics, and expressed security requirements as temporal properties over bigraphical predicates. We performed speculative threat analysis in order to reason about the consequences of the evolution of the environment on requirements satisfaction. To achieve this aim, we interpret the BRS over an LTS representing the evolution of the configuration of the CPSp. Subsequently, through model checking we identify all possible states in the evolution of the operational environment where security requirements are violated. The results of analysis are used during planning to generate an adaptation strategy consisting of security policies that the system can take to prevent, circumvent or mitigate violations. When, due to performance concerns, analysis can only be performed up to a maximum number of future states, validity of the adaptation strategy is bound to a lookahead horizon. In this case, we utilise a heuristic method to regenerate and enact the adaptation strategy at runtime. Our evaluation demonstrates that our approach can identify and counteract security requirements violations arising from changes in both constituents of a CPSp. Moreover, BRS are a suitable formalism to express such topological relationships. To evaluate our approach we automated the analysis and planning activities. We assessed the applicability of our approach and its overhead by using a case study formulated following discussions with an industrial partner concerned with countering insider threats. Our results are encouraging and provide evidence of the feasibility of the approach.

We have identified and are pursuing a number of promising avenues for further investigation. We plan to integrate our approach with more advanced techniques dealing with partial observation of environment events and decentralization of the enforcement controller. Additionally, we will relax the atomicity assumptions made regarding security policy enactment; in particular, we will encode metric temporal characteristics into the underlying logical framework used to deal with enforcement issues. This will also make our approach applicable to a wider set of physical environments that may exhibit inertia and will allow us to select an adaptative security policy plan that also depends on enactment timings. We are also considering incremental verification techniques to handle exogenous changes without invalidating previous analysis outcomes.

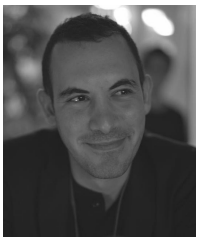
Acknowledgements

This work was partially supported by ERC Advanced Grants no. 227977 (SMScom) and no. 291652 (ASAP), and SFI grants 10/CE/I1855 and 13/RC/2094.

References

[1] E. A. Lee, "Cyber Physical Systems: Design Challenges," EECS Department, University of California, Berkeley, Tech. Rep., 2008.
[2] J. Depoy, J. Phelan, P. Sholander, B. Smith, G. Varnado, and G. Wyss, "Risk Assessment for Physical and Cyber Attacks on Critical Infrastructures," in *Proc. of the Military Communications Conference*, 2005, pp. 1961–1969.
[3] F. den Braber, I. Hogganvik, M. Lund, K. Stølen, and F. Vraalsen, "Model-Based Security Analysis in Seven Steps-A Guided Tour to the CORAS Method," *BT Technology Journal*, vol. 25, no. 1, 2007.
[4] ISO/IEC-27001/27005, "Information Technology. Security Techniques. (27001) Information Security Management Systems; (27005) Information Security Risk Management," 2008.
[5] A. van Cleeff, W. Pieters, R. Wieringa, and F. van Tiel, "Integrated Assessment and Mitigation of Physical and Digital Security Threats: Case Studies on Virtualization," *Inf. Sec. Techn. Report*, vol. 16, no. 3-4, pp. 142–149, 2011.
[6] L. Pasquale, C. Ghezzi, C. Menghi, C. Tsigkanos, and B. Nuseibeh, "Topology Aware Adaptive Security," in *Proc. of the 9th Int. Symp. on Software Engineering for Adaptive and Self-Managing Systems*, 2014, pp. 43–48.
[7] C. Tsigkanos, L. Pasquale, C. Menghi, C. Ghezzi, and B. Nuseibeh, "Engineering Topology Aware Adaptive Security: Preventing Requirements Violations at Runtime," in *Proc. of the 22nd Int. Requirements Engineering Conf.*, 2014, pp. 203–212.
[8] L. Cardelli and A. D. Gordon, "Mobile Ambients," in *Proc. of the 1st Int. Conf. on Foundations of Software Science and Computation Structure*, 1998, pp. 140–155.
[9] R. Milner, *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.
[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.
[11] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*. Prentice Hall Professional, 2003.
[12] A. van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proc. of the 26th Intl. Conference on Software Engineering*, 2004, pp. 148–157.
[13] H. Merz, T. Hansemann, and C. Hübner, *Building Automation: Communication Systems with EIB/KNX, LON and BACnet*. Springer Science & Business Media, 2009.
[14] W. Granzer, F. Praus, and W. Kastner, "Security in Building Automation Systems," *IEEE Trans. on Industrial Electronics*, vol. 57, no. 11, pp. 3622–3630, 2010.
[15] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.
[16] R. Milner, *Communicating and Mobile Systems: The Pi Calculus*. Cambridge University Press, 1999.
[17] S. Benford, M. Calder, T. Rodden, and M. Sevegnani, "On lions, impala, and bigraphs: Modelling interactions in physical/virtual spaces," *ACM Trans. Comput.-Hum. Interact.*, vol. 23, no. 2, pp. 9:1–9:56, May 2016.
[18] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT press, 1999.

[19] M. Sevegnani and M. Calder, "Bigraphs with Sharing," *Theor. Comput. Sci.*, vol. 577, pp. 43–73, 2015.
[20] G. Perrone, S. Debois, and T. T. Hildebrandt, "A Verification Environment for Bigraphs," *Innovations in Systems and Software Engineering*, vol. 9, no. 2, pp. 95–104, 2013.
[21] G. Bhat, R. Cleaveland, and O. Grumberg, "Efficient On-the-Fly Model Checking for CTL," in *Proc. of the 10th Symposium on Logic in Computer Science*, 1995, pp. 388–397.
[22] A. A. E. Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin, "Organization Based Access Control," in *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks*, 2003, pp. 120–131.
[23] F. B. Schneider, "Enforceable Security Policies," *ACM Transactions on Information and System Security*, vol. 3, no. 1, pp. 30–50, 2000.
[24] C. Tsigkanos, L. Pasquale, C. Ghezzi, and B. Nuseibeh, "Ariadne: Topology aware adaptive security for cyber-physical systems," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, 2015.
[25] L. A. Walton and M. Worboys, "A qualitative bigraph model for indoor space," in *Geographic Information Science*. Springer, 2012.
[26] C. Tsigkanos, T. Kehrer, C. Ghezzi, L. Pasquale, and B. Nuseibeh, "Adding static and dynamic semantics to building information models," in *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*. ACM, 2016, pp. 1–7.
[27] E. Pereira, C. Kirsch, and R. Sengupta, "BiAgents–A Bigraphical Agent Model for Structure-aware Computation," *Cyber-Physical Cloud Computing Working Papers, CPCC Berkeley*, 2012.
[28] M. Calder, A. Koliousis, M. Sevegnani, and J. Sventek, "Real-time verification of wireless home networks using bigraphs with sharing," *Science of Computer Programming*, vol. 80, 2014.
[29] M. Sevegnani, C. Unsworth, and M. Calder, "A SAT Based Algorithm for the Matching Problem in Bigraphs with Sharing," University of Glasgow, Tech. Rep., 2010.
[30] E. Højsgaard and A. J. Glenstrup, "The BPL Tool: A Tool for Experimenting with Bigraphical Reactive Systems," *Bigraphical Languages and their Simulation*, p. 85, 2011.
[31] L. Birkedal, T. C. Damgaard, A. J. Glenstrup, and R. Milner, "Matching of Bigraphs," *Electronic Notes in Theoretical Computer Science*, vol. 175, no. 4, pp. 3–19, 2007.
[32] G. Bacci, D. Grohmann, and M. Miculan, "DBtk: A Toolkit for Directed Bigraphs," in *Algebra and Coalgebra in Computer Science*. Springer, 2009, pp. 413–422.
[33] J. Krivine, R. Milner, and A. Troina, "Stochastic bigraphs," *Electronic Notes in Theoretical Computer Science*, vol. 218, 2008.
[34] M. Krotofil and A. A. Cárdenas, "Resilience of Process Control Systems to Cyber-Physical Attacks," in *Proc. of the 18th Nordic Conference on Secure IT Systems*, 2013, pp. 166–182.
[35] S. Sridhar and M. Govindarasu, "Model-Based Attack Detection and Mitigation for Automatic Generation Control," *IEEE Trans. on Smart Grid*, vol. 5, no. 2, pp. 580–591, 2014.
[36] R. Akella, H. Tang, and B. M. McMillin, "Analysis of Information Flow Security in CyberPhysical Systems," *Int. J. of Critical Infrastructure Protection*, vol. 3, no. 34, pp. 157 – 173, 2010.
[37] R. Focardi and R. Gorrieri, "The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties," *IEEE Trans. Software Eng.*, vol. 23, no. 9, pp. 550–571, 1997.
[38] C. Piazza, E. Pivato, and S. Rossi, "CoPS - Checker of Persistent Security," in *Proc. of the 10th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, 2004, pp. 144–152.
[39] M. Burmester, E. Magkos, and V. Chrissikopoulos, "Modeling Security in Cyber-Physical Systems," *Int. J. of Critical Infrastructure Protection*, vol. 5, no. 34, pp. 118 – 126, 2012.
[40] T. Dimkov, W. Pieters, and P. Hartel, "Portunes: Representing Attack Scenarios Spanning Through the Physical, Digital and Social Domain," in *Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, 2011, pp. 112–129.
[41] R. De Nicola, G. L. Ferrari, and R. Pugliese, "KLAIM: A Kernel Language for Agents Interaction and Mobility," *IEEE Trans. on Software Engineering*, vol. 24, no. 5, pp. 315–330, 1998.
[42] M. S. Kirkpatrick, M. L. Damiani, and E. Bertino, "Prox-RBAC: a Proximity-Based Spatially Aware RBAC," in *Proc. of the 19th Int. Conference on Advances in Geographic Information Systems*, 2011, pp. 339–348.
[43] A. Gupta, M. S. Kirkpatrick, and E. Bertino, "A Formal Proximity Model for RBAC Systems," *Computers & Security*, vol. 41, pp. 52–67, 2014.
[44] E. Yuan, N. Esfahani, and S. Malek, "A Systematic Survey of Self-Protecting Software Systems," *ACM Trans. on Autonomous and Adaptive Systems*, vol. 8, no. 4, p. 17, 2014.
[45] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh, "Requirements-Driven Adaptive Security: Protecting Variable Assets at Runtime," in *Proc. of the 20th Int. Requirements Engineering Conf.*, 2012, pp. 111–120.
[46] E. Yuan, S. Malek, B. R. Schmerl, D. Garlan, and J. Gennari, "Architecture-Based Self-Protecting Software Systems," in *Proc. of the 9th Int. Conf. on Quality of Software Architectures*, 2013, pp. 33–42.
[47] D. Basin, V. Jugé, F. Klaedtke, and E. Zălinescu, "Enforceable security policies revisited," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 1, p. 3, 2013.
[48] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.
[49] J. Ligatti, L. Bauer, and D. Walker, "Edit Automata: Enforcement Mechanisms for Run-Time Security Policies," *International Journal of Information Security*, vol. 4, no. 1-2, pp. 2–16, 2005.



Christos Tsigkanos received a BSc degree in computer science from University of Athens and a MSc degree in software engineering from University of Amsterdam. He is working towards his PhD in adaptive security under prof. Carlo Ghezzi at Politecnico di Milano. His research interests lie in the intersection of software engineering and security, and include self-adaptive systems, cyber-physical systems, requirements engineering and formal verification.

Liliana Pasquale is a research fellow at Lero - the Irish Software Research Centre (Ireland) since April 2015. She received her PhD in Information and Communication Technology from Politecnico di Milano in 2011. Her research interests are in requirements engineering and adaptive systems, with particular focus on security, privacy and digital forensics.

Carlo Ghezzi is Professor of Software Engineering in Politecnico di Milano. He is an ACM Fellow, an IEEE Fellow, a member of the European Academy of Sciences and the Italian Academy of Sciences. He received the ACM SIGSOFT Outstanding Research Award and the Distinguished Service Award. He is the current President of Informatics Europe. He has been the Editor-in-Chief of the ACM Trans. on Software Engineering and Methodology and is currently an Associate Editor of the Communications of the ACM, IEEE Trans. on Software Engineering, Science of Computer Programming, Computing, and Service Oriented Computing and Applications. His research has been mostly focusing on different aspects of software engineering. He co-authored over 200 papers and 8 books. He coordinated several national and international research projects.

Bashar Nuseibeh is Professor of Computing at The Open University (Director of Research 2001-2008) and Professor of Software Engineering at Lero - the Irish Software Research Centre (Chief Scientist 2009-2012). He is a Visiting Professor at Imperial College London and the National Institute of Informatics, Japan. His research interests lie at the intersection of requirements engineering, adaptive systems, and security and privacy. He served as Editor-in-Chief of IEEE Trans. on Software Engineering and the Automated Software Engineering Journal. He received an ICSE Most Influential Paper Award, a Philip Leverhulme Prize, an Automated Software Engineering Fellowship, a Senior Research Fellowship of the Royal Academy of Engineering, and an ACM SIGSOFT Distinguished Service Award. He currently holds a Royal Society-Wolfson Merit Award and an ERC Advanced Grant on Adaptive Security and Privacy.